02

¿Qué es un pool y un datasource?

Transcripción

[00:00] Hola, ¿cómo les va? Antes de seguir con los nuevos contenidos del curso, vamos a hacer una revisión de cómo estamos con el proyecto. Nosotros estamos conectando nuestra aplicación con la base de datos MySQL. Aquí tenemos MySQL y tenemos nuestra aplicación Java. Y esas tienen una conexión.

[00:26] Pero nosotros aprendimos que estas dos partes no se conectan directamente. Voy a aumentar un poquito acá. Estas dos partes no se conecten directamente. ¿Por qué? Porque una es Java y la otra parte es una base de datos que tiene su propio protocolo de comunicación y hacerlo directamente no es tan sencillo así.

[00:50] Para resolver eso, nosotros agregamos un driver a nuestra aplicación, entonces tenemos acá un driver MySQL que nos facilita esta conexión, entonces desde el driver nosotros logramos hacer nuestra conexión a la base de datos. Y para cada base de datos tenemos un driver específico.

[01:16] Ahora nosotros ya estamos utilizando el MySQL, pero si nosotros cambiamos la base para el SQL Server, nosotros también tenemos que cambiar el driver de la aplicación. Y en lugar de ir al MySQL, estamos yendo a driver de SQL Server. Y eso cambia todo otra vez. Si hacemos eso, tenemos que salir modificando todo el código de la aplicación para cambiar las interfaces específicas de cada driver, como vimos en el inicio del curso.

[01:52] Para resolver esta situación, nosotros tenemos una interfaz llamada JDBC, que desde sus clases e interfaces logramos acceder a la implementación de los drivers de forma transparente, entonces con el JDBC acá, no importa para nosotros qué base de datos estamos utilizando, nosotros vamos a lograr conectarnos a ellos porque tenemos las interfaces y queda todo lo que es específico, queda transparente para nosotros.

[02:27] Bueno, con eso nosotros no sufrimos más cuando cambiamos la base de datos, entonces no importa qué base de datos es, con el JDBC nos conectamos acá con el MySQL y está todo bien. Avancemos un poco más. Avanzamos con el desarrollo y vimos que siempre repetimos la rutina de abrir una conexión con la base de datos.

[02:52] Enviamos nuevas informaciones de URL de conexión, usuario y contraseña, todas las veces. Si agregamos un nuevo método, una nueva clase que se conectaba a la base de datos, teníamos que replicar esta lógica y otra vez tendríamos un trabajo gigante de mantenimiento por culpa de esta replicación de código.

[03:14] Para resolver eso y facilitar aún más nuestro desarrollo, nosotros realizamos la refactorización del proyecto y acá nosotros utilizamos el patrón de diseño factory para crear la clase connectionFactory. Tenemos acá ahora connectionFactory, ¿qué hace? Ella centraliza esta lógica de apertura de una conexión con la base de datos.

[03:39] Entonces nosotros no tenemos más que estar replicando esta lógica para cada método, cada clase que estamos utilizando. Tenemos todo ahí en esta connectionFactory. Entonces entra un componente más a nuestra aplicación, acá el connectionFactory que hace este medio de campo acá para nosotros. Para arreglar un poquito más esto de acá voy a ponerlo en el tope, ahora sí.

[04:09] Y acá la connectionFactory que nos facilita eso. Listo, acá está la base de nuestro proyecto hoy. Ahora analicemos la siguiente situación. Todas las

requisiciones que realizamos hoy en la aplicación hasta la base de datos son de una única conexión.

[04:26] Para cada operación que realizamos en la pantalla, la aplicación abre la conexión con el connectionFactory, realiza la operación en la base de datos y la cierra al final, pero cambiemos un poco el contexto y vayamos acá a la página de Alura. Acá en la página de Alura tenemos muchas categorías de cursos.

[04:52] Si hacemos un clic acá en cursos, tenemos las categorías, y cuando elijo una categoría acá voy a esta pantalla y ella muestra todos los cursos que están disponibles acá. Y supongamos que cada información de esta que tenemos en la pantalla está guardada en una base de datos.

[05:09] Y cada clic que hago acá para ver más informaciones del curso, una conexión con la base de datos es abierta. Entonces, cuando entro acá al curso de lógica de programación, se abre una nueva conexión, pero aquí en la página de Alura, al mismo tiempo que estoy accediendo a las páginas, hay otras personas que también están haciendo lo mismo.

[05:33] ¿Y cómo funcionaría esto en nuestra aplicación? Si realizamos una requisición y otra persona también la hace, ¿va a haber una cola para que la próxima requisición sea procesada al final de la primera? No tiene sentido, imagina el tamaño de la cola con miles de usuarios intentando acceder a las páginas de los cursos de Alura y esperando una eternidad.

[05:54] Entonces nuestra aplicación debe estar disponible para que más personas puedan utilizarla al mismo tiempo, pudiendo utilizar más conexiones con la base de datos en paralelo. Pero todo eso tiene un límite. ¿Hasta cuántas conexiones una base de datos puede atender? No importa cuál base de datos estamos utilizando, si hay es MySQL, Postgre SQL, SQL Server.

[06:17] Todas las bases de datos tienen un límite y pueden caer si llegan a eso. ¿Entonces cómo podemos solucionar esta situación y lograr atender a múltiples usuarios sin ahogar a nuestras bases de datos? La mejor alternativa

acá es implementar una solución que pueda disponibilizar una cantidad equis de conexiones abiertas para atender a las requisiciones paralelas que ocurran.

[06:41] Pero nosotros no podemos dejar que la aplicación siga abriendo conexiones sin límite, entonces debemos poder configurar una cantidad mínima, una cantidad máxima de conexiones. Esta implementación va a comunicarse con el JDBC para abrir la cantidad establecida que nosotros configuramos de conexiones y listo.

[07:03] Entonces, acá en nuestro dibujo nosotros vamos a agrandar un poquito más acá porque vamos a tener un nuevo elemento acá. Entra acá una nueva cajita que va a estar ayudándonos a manejar esta cantidad de conexiones que están recreadas ¿Y esta cajita quién la implementa? ¿Nosotros? No. Por suerte, nosotros no necesitamos implementar esta funcionalidad porque hay un montón de librerías que ya tienen a esta necesidad.

[07:31] Incluso estas soluciones, conocida como pool de conexiones. Entonces aquí vamos a poner un nombre, pool de conexiones. Ahí está su nombre. Voy a subir el nombre acá para que quede más lindo. Y ahora sí.

[07:51] O sea, tenemos un pool de conexiones que va a comunicarse con el JDBC para mantener una cantidad mínima y una cantidad máxima de conexiones abiertas en la aplicación para que pueda atender a las requisiciones sin que tenga una cola muy grande y que no ahogue la aplicación.

[08:09] Tal como los drivers de base de datos, también hay varias opciones para drivers que manejen un pool de conexiones. El C3P0, acá voy a agregar, C3P0 es una de las opciones más conocidas en el mercado. Y su implementación es bastante sencilla, cómo vamos a ver más adelante en el curso.

[08:30] Para utilizarlo bien vamos a aprovechar las ventajas del JDBC para utilizar una nueva interfaz llamada datasource. Esta interfaz va abstraer la implementación del pool de conexiones para nosotros y con esta nueva

estructura, la connectionFactory que tenemos no va más a tener la responsabilidad de crear una conexión para nosotros.

[08:52] Lo que ella va a hacer ahora es solicitar una conexión abierta para el data source, entonces acá esta cajita del pool de conexiones va a estar adentro de una super caja y acá yo voy a agrandar un poquito más el dibujo, así nosotros no nos perdemos, pero vamos a tener acá una super caja en donde este pool de conexiones del C3P0 va a estar acá dentro.

[09:23] Para traerlo, lo envío hacia atrás, ahora sí. Esto acá también. Ahí tenemos. Y esta super caja, como había dicho, va a llamarse datasource. Está en el tope, ahí está. Aquí adentro tenemos el pool de conexiones de C3P0, que es el que vamos a utilizar y acá dentro vamos a tener varias, pero varias pelotitas.

[09:57] Vamos con las pelotitas acá que son nuestras conexiones de base de datos que tenemos acá, todas preparadas con el mínimo y el máximo que podemos llegar, entonces acá va a estar siempre con el mínimo y cuando utilicemos todas, van a ser creadas más hasta que llegue al tope que configuremos.

[10:20] Y ahí el connection Factory va a solicitar el datasource y va a decir en lugar de la connectionFactory decir: "JDBC, dame una nueva conexión". El connectionFactory va a decir: "datasource, dame una conexión que tienes ahí en tu en tu pool de conexiones y yo sigo acá con mi trabajo", y el datasource es el que va a manejar toda esta parte de le doy una conexión nueva de una conexión que ya tengo acá creada, veo acá con mis configuraciones cómo esto va a ser manejado.

[10:53] De esta forma, nosotros no tenemos nuestra aplicación con una configuración sostenible de conexiones abiertas, así no tenemos ningún problema de performance ni de agotamiento de recursos, tal como las aplicaciones reales del mercado, y por ahora es eso.

[11:13] En las próximas clases vamos a tener un poco más de acción, implementando la utilización de esta nueva dependencia del pool de conexiones. Vamos a ver cómo funciona el control del pool de conexiones y cómo funciona una implementación de la interfaz de datasource.